

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ
ВЫСШЕГО ОБРАЗОВАНИЯ
«САХАЛИНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»
ПОЛИТЕХНИЧЕСКИЙ КОЛЛЕДЖ

МЕТОДИЧЕСКИЕ УКАЗАНИЯ

**для студентов по выполнению
практических работ
по дисциплине**

ПМ.03 Участие в разработке приложений взаимодействия с интеллектуальными интегрированными системами

**укрупненная группа:
09.00.00 ИНФОРМАТИКА И ВЫЧИСЛИТЕЛЬНАЯ ТЕХНИКА
специальность: 09.02.08 Интеллектуальные
интегрированные системы**

базовый уровень подготовки

**Квалификация: Техник по интеллектуальным интегрированным системам
Форма обучения: очная**

Южно-Сахалинск
2025

УТВЕРЖДАЮ
Зам. директора по НМР
 Е.Н. Ермолаева
«24» марта 2025 г.

Разработчик: Перов И.А., преподаватель информационных дисциплин

Одобрено на заседании ПЦК
информационных дисциплин
Протокол № 8 от «21» марта 2025г.
Председатель ПЦК



О.Б. Куприяшина

Оглавление

1. ОБЩИЕ ПОЛОЖЕНИЯ	2
2. ПРАКТИЧЕСКИЕ РАБОТЫ (РАЗДЕЛ 1. СЕТЕВЫЕ И ОБЛАЧНЫЕ ТЕХНОЛОГИИ)...	3
Практическая работа №1-2. Знакомство с Packet Tracer. Моделирование простой сети – 4 часа.....	3
Практическая работа №3-4. Обеспечение информационной безопасности сетей – 4 часа .	3
Практическая работа №5-6. Система создания и конфигурирования виртуальной среды разработки – 4 часа	4
Практическая работа №7. Конфигурирование виртуальной среды (в выбранной среде) – 2 часа.....	5
Практическая работа №8. Знакомство с облачной платформой – 2 часа	6
Практическая работа №9-10. Знакомство с сервисами в облачной платформе – 4 часа	7
Практическая работа №11-12. Создание приложения для облачной платформы – 4 часа	7
3. ПРАКТИЧЕСКИЕ РАБОТЫ (РАЗДЕЛ 2. РАЗРАБОТКА ПРИЛОЖЕНИЙ УПРАВЛЕНИЯ ИНТЕГРИРОВАННЫМИ СИСТЕМАМИ)	9
Практическая работа №1. Выполнение конфигурационных работ с микроконтроллером – 4 часа (ориентировочно)	9
Практическая работа №2. Первичная настройка проекта под микроконтроллер – 4 часа	9
Практическая работа №3. Работа с программным кодом (анализ и отслеживание изменений) – 2-4 часа.....	10
Практическая работа №4. Работы по настройке программных инструментов под микроконтроллер – 4 часа	11
Практическая работа №5. Настройка программатора для прошивки микроконтроллера – 4 часа.....	12
Практическая работа №6. Анализ программного кода для микроконтроллера – 4 часа.	12
Практическая работа №7. Написание приложения для опроса датчиков и отправки значений через MQTT – 4 часа	13
4. КРИТЕРИИ ОЦЕНКИ ПРАКТИЧЕСКИХ РАБОТ	15
5. ЛИТЕРАТУРА И ПРОГРАММНОЕ ОБЕСПЕЧЕНИЕ.....	16

1. ОБЩИЕ ПОЛОЖЕНИЯ

Цели практических работ:

- Научиться моделировать сети в Cisco Packet Tracer, настраивать IP-адресацию, маршрутизацию и безопасность.
- Освоить работу с облачными платформами (IaaS, PaaS, SaaS), создание и развертывание приложений в облаке.
- Приобрести навыки конфигурирования микроконтроллеров (STM32, AVR) в среде IDE (STM32CubeIDE, Keil, Arduino IDE), написания и отладки программ, работы с периферией и интерфейсами.
- Освоить инструменты контроля версий (Git), анализа кода, профилирования и прошивки устройств.

Оборудование и ПО:

- **Для сетей и облаков:** компьютер с ОС Windows/Linux, Cisco Packet Tracer (версия 8.x), VirtualBox/VMware, браузер с доступом к облачным платформам (Yandex Cloud, Google Cloud, AWS – по выбору).
- **Для микроконтроллеров:** плата STM32F103 (Blue Pill) или Arduino Uno, программатор ST-Link/USB-UART, среда разработки STM32CubeIDE или Arduino IDE, датчики (температуры, движения, ультразвуковой), двигатели, модули связи (ESP8266, nRF24L01).

Техника безопасности:

При работе с макетными платами и микроконтроллерами не превышать напряжение питания (3.3В/5В). Перед сборкой цепей отключать питание.

2. ПРАКТИЧЕСКИЕ РАБОТЫ (РАЗДЕЛ 1. СЕТЕВЫЕ И ОБЛАЧНЫЕ ТЕХНОЛОГИИ)

Практическая работа №1-2. Знакомство с Packet Tracer. Моделирование простой сети – 4 часа

Цель: освоить интерфейс Cisco Packet Tracer, создать и настроить простую локальную сеть из двух ПК и коммутатора, проверить связность.

Оборудование: ПК с Packet Tracer.

Задание:

1. Запустить Packet Tracer. Изучить панели устройств, соединений, симуляции.
2. Разместить на рабочем поле:
 - 2 компьютера (PC-PT);
 - 1 коммутатор (Switch 2960).
3. Соединить каждый компьютер с коммутатором медным прямым кабелем (Copper Straight-Through).
4. Назначить IP-адреса: PC1 – 192.168.1.1/24, PC2 – 192.168.1.2/24. Шлюз по умолчанию не указывать.
5. Проверить связь командой `ping` из симуляции или из командной строки ПК.
6. Перейти в режим симуляции (Simulation), запустить `ping`, просмотреть PDU (пакеты) на каждом этапе (ARP, ICMP).
7. Сохранить проект.

Методические указания:

- Для назначения IP: кликнуть на ПК → Desktop → IP Configuration.
- Для проверки: Desktop → Command Prompt → `ping 192.168.1.2`.
- В режиме симуляции можно пошагово наблюдать кадры Ethernet.

Форма отчёта:

Скриншот топологии, таблица IP-адресов, результат `ping`, описание прохождения пакета (какой MAC-адрес, какой коммутатор узнал).

Контрольные вопросы:

1. Как коммутатор узнаёт MAC-адреса?
2. Зачем нужен протокол ARP?
3. Что произойдёт, если у двух ПК одинаковый IP?

Практическая работа №3-4. Обеспечение информационной безопасности сетей – 4 часа

Цель: настроить базовые средства защиты: списки контроля доступа (ACL), порты безопасности на коммутаторе, SSH вместо Telnet.

Оборудование: Packet Tracer, маршрутизатор (Router 2811), коммутатор, два ПК.

Задание:

1. Собрать сеть: маршрутизатор R1, коммутатор SW1, PC1, PC2. Назначить IP: PC1 – 192.168.1.10/24, PC2 – 192.168.1.20/24, интерфейс маршрутизатора Fa0/0 – 192.168.1.1/24.
2. Настроить на маршрутизаторе:
 - пароль на консоль и VTY (line console 0, line vty 0 4);
 - включить SSH (hostname, ip domain-name, crypto key generate rsa);
 - создать локального пользователя admin;
 - запретить Telnet, разрешить SSH.
3. Настроить стандартный ACL (номер 1), запрещающий PC1 доступ к маршрутизатору по ICMP (ping), но разрешающий остальной трафик. Применить ACL на интерфейс Fa0/0 входящим (in).
4. На коммутаторе настроить port security на порту, подключенном к PC1: максимальное количество MAC-адресов – 1, нарушение – shutdown.
5. Проверить: ping от PC1 к маршрутизатору должен быть заблокирован, от PC2 – разрешен. Попробовать подключить дополнительное устройство к порту PC1 – порт должен отключиться.

Методические указания:

- Использовать CLI маршрутизатора.
- Пример ACL: `access-list 1 deny host 192.168.1.10`, `access-list 1 permit any`, затем `interface fa0/0`, `ip access-group 1 in`.
- Port security: `switchport port-security`, `switchport port-security maximum 1`, `switchport port-security violation shutdown`.

Форма отчёта:

Скриншоты конфигураций, вывод `show access-lists`, `show port-security`, результаты ping.

Контрольные вопросы:

1. Чем отличаются стандартные и расширенные ACL?
2. Как восстановить порт после нарушения port security?
3. Почему SSH безопаснее Telnet?

Практическая работа №5-6. Система создания и конфигурирования

виртуальной среды разработки – 4 часа

Цель: установить и настроить виртуальную машину (VirtualBox) с ОС Linux (Astra или Ubuntu) для разработки приложений.

Оборудование: ПК с VirtualBox, ISO-образ Astra Linux Common Edition (или Ubuntu).

Задание:

1. Создать новую виртуальную машину:

- тип ОС: Linux, версия: Debian (64-bit);
- ОЗУ: 2 ГБ; диск: VDI, динамический, 20 ГБ.
- 2. Загрузить ISO, установить ОС (русский язык, имя пользователя dev, пароль 123456).
- 3. Установить гостевые дополнения (Guest Additions) для возможности общего буфера обмена и изменения разрешения экрана.
- 4. Настроить сеть: переключить на «Мост» (Bridged) или NAT, проверить доступ в интернет (ping ya.ru).
- 5. Установить необходимые инструменты разработки: git, gcc, make, python3, vs code (или любой редактор).
- 6. Создать пользователя с правами sudo (если не создан).

Методические указания:

- После установки ОС выполнить `sudo apt update && sudo apt upgrade`.
- Установка гостевых дополнений: меню VirtualBox → Devices → Insert Guest Additions CD image, затем в терминале запустить `sudo ./VBoxLinuxAdditions.run`.

Форма отчёта:

Скриншоты процесса установки, настройки сети, вывода `ifconfig` или `ip a`, список установленного ПО.

Контрольные вопросы:

1. Чем отличается NAT от Bridged в VirtualBox?
2. Зачем нужны гостевые дополнения?
3. Как сделать снимок состояния (snapshot) виртуальной машины?

Практическая работа №7. Конфигурирование виртуальной среды (в выбранной среде) – 2 часа

Цель: настроить дополнительные параметры виртуальной среды: общие папки, изолированную сеть, клонирование.

Оборудование: VirtualBox, виртуальная машина из предыдущей работы.

Задание:

1. Настроить общую папку между хостовой и гостевой ОС (например, `C:\share` на хосте → `/mnt/share` в гостевой).
2. Создать ещё одну виртуальную машину (клон) с именем `dev-clone` (полное клонирование).
3. Настроить между двумя ВМ внутреннюю сеть (Internal Network) с именем `intnet`. Назначить IP-адреса вручную (192.168.100.1/24 и 192.168.100.2/24).
4. Проверить связь `ping` между ВМ.
5. Настроить проброс портов (NAT) для доступа из хостовой ОС к веб-серверу на гостевой (например, порт 8080 хост → 80 гость).

Методические указания:

- Для общей папки: в настройках VM → Shared Folders → добавить папку, указать точку монтирования и флаг Auto-mount.
- Для внутренней сети: в настройках сетевого адаптера выбрать «Internal Network», задать имя.
- Проброс портов: в настройках NAT → Advanced → Port Forwarding.

Форма отчёта:

Скриншоты настройки общей папки, результатов ping между VM, проверки проброса портов.

Контрольные вопросы:

1. Как смонтировать общую папку вручную через mount?
2. Чем отличается внутренняя сеть от сетевого моста?
3. Для чего используется клонирование VM?

Практическая работа №8. Знакомство с облачной платформой – 2 часа

Цель: зарегистрироваться в облачной платформе (Yandex Cloud или Google Cloud), изучить консоль управления и создать первый проект.

Оборудование: компьютер с браузером, доступ в интернет, банковская карта (для активации пробного периода, если требуется).

Задание:

1. Выбрать облачную платформу (рекомендуется Yandex Cloud – есть бесплатный грант на 4000 ₽).
2. Зарегистрироваться, создать платёжный аккаунт (указать карту, но списания не будет при соблюдении лимитов).
3. Изучить интерфейс консоли: перечень сервисов (Compute Cloud, Object Storage, VPC, IAM).
4. Создать сервисный аккаунт с ролью editor.
5. Создать виртуальную машину (Instance) с параметрами: 1 vCPU, 2 ГБ RAM, образ Ubuntu 20.04, публичный IP.
6. Подключиться к VM по SSH (используя консольный SSH-клиент или PuTTY).
7. Удалить VM (чтобы не тратить средства).

Методические указания:

- В Yandex Cloud для подключения по SSH можно сгенерировать ключ в консоли.
- После создания VM подождать 2-3 минуты, затем скопировать её публичный IP.
- Команда подключения: `ssh -i ключ.pem user@ip`.

Форма отчёта:

Скриншоты регистрации, создания VM, подключения по SSH (вывод `whoami`), удаления.

Контрольные вопросы:

1. Что такое IAM и сервисный аккаунт?
2. Как ограничить расходы в облаке?
3. Чем отличается виртуальная машина от контейнера?

Практическая работа №9-10. Знакомство с сервисами в облачной платформе –

4 часа

Цель: изучить основные сервисы IaaS, PaaS, SaaS на примере Yandex Cloud: Object Storage, Managed Databases, Cloud Functions.

Оборудование: браузер, доступ к облачной консоли.

Задание:

1. **Object Storage:** создать бакет (bucket) с уникальным именем, загрузить в него любой файл (например, изображение), сделать его публичным, получить ссылку.
2. **Managed Databases:** создать кластер PostgreSQL (однохостовой, минимальная конфигурация), подключиться к нему через утилиту `psql` из облачной ВМ или локально, создать таблицу `test`, вставить строку.
3. **Cloud Functions:** написать простую функцию на Python, которая возвращает "Hello, Cloud!", опубликовать её, вызвать через API-шлюз.
4. Просмотреть логи и метрики каждой службы.

Методические указания:

- Для доступа к PostgreSQL используйте предварительно созданную ВМ (из предыдущей работы) или установите `psql` локально.
- Для Cloud Functions можно использовать встроенный редактор кода в консоли.

Форма отчёта:

Скриншоты бакета с файлом, результаты запроса к БД (`SELECT *`), текст функции и результат её вызова.

Контрольные вопросы:

1. В чем разница между Object Storage и блочным хранилищем?
2. Как обеспечивается отказоустойчивость Managed Databases?
3. Какие триггеры можно использовать для Cloud Functions?

Практическая работа №11-12. Создание приложения для облачной платформы

– 4 часа

Цель: разработать и развернуть простое веб-приложение (например, API для управления светодиодом) в облаке с использованием PaaS.

Оборудование: облачная платформа, среда разработки (Python/Flask или Node.js).

Задание:

1. Написать приложение на Python с использованием Flask, которое:
 - принимает GET-запрос по пути / и возвращает "OK";
 - принимает POST-запрос /led с JSON {"state": "on"} и логирует состояние;
 - имеет встроенный веб-сервер (WSGI).
2. Упаковать приложение в Docker-контейнер (написать Dockerfile).
3. Загрузить образ в реестр контейнеров (Container Registry).
4. Развернуть контейнер в сервисе Serverless Containers (Yandex Cloud) или Cloud Run (Google Cloud), указав публичный доступ.
5. Протестировать API с помощью curl или Postman.

Методические указания:

- Flask: `app = Flask(__name__)`, `app.run(host='0.0.0.0', port=8080)`.
- Dockerfile: `FROM python:3.9-slim`, `COPY . /app`, `RUN pip install flask`, CMD.
- Для развертывания использовать инструкции облачного провайдера.

Форма отчёта:

Листинг кода приложения, Dockerfile, команды сборки и развертывания, URL для проверки, скриншоты ответов curl.

Контрольные вопросы:

1. Что такое контейнеризация и её преимущества перед виртуальными машинами?
2. Как настроить автомасштабирование в Serverless Containers?
3. Как защитить API с помощью API-ключа?

3. ПРАКТИЧЕСКИЕ РАБОТЫ (РАЗДЕЛ 2. РАЗРАБОТКА ПРИЛОЖЕНИЙ УПРАВЛЕНИЯ ИНТЕГРИРОВАННЫМИ СИСТЕМАМИ)

Практическая работа №1. Выполнение конфигурационных работ с микроконтроллером – 4 часа (ориентировочно)

Цель: научиться настраивать периферию микроконтроллера (тактовую частоту, пины, таймеры) с помощью графического инструмента (STM32CubeMX или Arduino IDE).

Оборудование: STM32F103 (Blue Pill) или Arduino Uno, ST-Link программатор, USB-кабель, STM32CubeIDE / Arduino IDE.

Задание:

1. Создать проект в STM32CubeIDE для микроконтроллера STM32F103C8T6.
2. С помощью CubeMX настроить:
 - тактовую частоту: HSE = 8 MHz (внешний кварц), PLL, получить SYSCLK = 72 MHz;
 - пин PC13 как выход (GPIO_Output) – на плате Blue Pill это встроенный светодиод;
 - таймер TIM2 в режиме генерации прерываний каждые 1000 мс.
3. Сгенерировать код и в функции main добавить мигание светодиодом с использованием таймера (без HAL_Delay).
4. Для Arduino: настроить пины 13 как выход, использовать таймер Timer1 (библиотека TimerOne) для прерываний.

Методические указания:

- В CubeMX: выбрать микроконтроллер, закладка Clock Configuration, установить HSE, PLL.
- В коде: HAL_GPIO_TogglePin(GPIOC, GPIO_PIN_13); в обработчике прерывания таймера.

Форма отчёта:

Скриншоты настроек CubeMX (такты, пины), листинг кода (обработчик таймера), видео мигания светодиода.

Контрольные вопросы:

1. Почему для STM32 важно правильно настроить тактирование?
2. Чем отличается таймер в режиме прерываний от polling?
3. Какую максимальную частоту ШИМ можно получить на TIM2?

Практическая работа №2. Первичная настройка проекта под микроконтроллер – 4 часа

Цель: настроить систему сборки, подключить необходимые библиотеки, настроить отладчик.

Оборудование: STM32CubeIDE, отладчик ST-Link, плата.

Задание:

1. Создать новый проект, настроить отладчик (ST-Link, SWD).
2. Добавить в проект библиотеку для работы с UART (стандартная HAL).
3. Настроить USART1 на скорости 115200 бод, пины PA9 (TX), PA10 (RX).
4. Написать функцию `printf` через UART (ретрансляция `_write`).
5. Проверить вывод в терминал (PuTTY или Arduino Serial Monitor).

Методические указания:

- Для редиректа `printf`: переопределить `int __io_putchar(int ch)`, отправляющий байт через UART.
- В main: `printf("Hello STM32\r\n");` зациклить.

Форма отчёта:

Скриншот настроек UART в CubeMX, листинг `__io_putchar`, скриншот вывода в терминале.

Контрольные вопросы:

1. Как работает функция `HAL_UART_Transmit`?
2. Зачем нужна синхронизация при отправке через UART?
3. Как изменить скорость UART без перекомпиляции?

Практическая работа №3. Работа с программным кодом (анализ и отслеживание изменений) – 2-4 часа

Цель: освоить систему контроля версий Git для отслеживания изменений в прошивке микроконтроллера.

Оборудование: компьютер с Git, репозиторий на GitHub/GitLab, проект из предыдущей работы.

Задание:

1. Установить Git, настроить `user.name` и `user.email`.
2. Создать локальный репозиторий в папке проекта микроконтроллера.
3. Добавить файлы (`.c`, `.h`, `.ioc`) в репозиторий, сделать первый коммит.
4. Изменить код: добавить второй светодиод на другой пин, скомпилировать, проверить.
5. Сделать второй коммит.
6. Создать ветку `feature/uart`, переключиться, добавить отправку температуры по UART, закоммитить.
7. Слить ветку с `main` (merge).
8. Отправить репозиторий на удалённый сервер (GitHub).

Методические указания:

- Не добавлять в Git временные файлы сборки (Debug/, *.o, *.elf) – использовать .gitignore.
- Пример .gitignore для STM32CubeIDE: /Debug/, *.launch, *.d.

Форма отчёта:

Скриншоты вывода `git log --oneline --graph`, ссылка на удалённый репозиторий.

Контрольные вопросы:

1. Чем отличается `git merge` от `git rebase`?
2. Как откатить коммит?
3. Что такое `git stash`?

Практическая работа №4. Работы по настройке программных инструментов под микроконтроллер – 4 часа

Цель: настроить инструменты статического анализа кода (Cppcheck) и профилирования (Perf или простые таймеры).

Оборудование: ПК с установленным Cppcheck, проект МК.

Задание:

1. Установить Cppcheck (бесплатный анализатор C/C++).
2. Запустить анализ проекта: `cppcheck --enable=all --suppress=missingIncludeSystem ..`
3. Исправить найденные предупреждения (например, неинициализированные переменные, утечки памяти).
4. Добавить в код профилирование времени выполнения цикла с помощью SysTick или TIM2: замерить время выполнения критической функции.
5. Вывести результат профилирования в UART.

Методические указания:

- Для замера времени: сохранить `HAL_GetTick()` до и после выполнения, разность в мс.
- Для точных измерений использовать TIM2 в режиме захвата.

Форма отчёта:

Отчёт Cppcheck до и после исправлений, листинг кода профилирования, результат в терминале.

Контрольные вопросы:

1. Какие классы ошибок выявляет статический анализатор?
2. Как измерить время выполнения в микросекундах?
3. Что такое «ложное срабатывание» анализатора?

Практическая работа №5. Настройка программатора для прошивки микроконтроллера – 4 часа

Цель: освоить утилиты командной строки для прошивки (OpenOCD, st-flash, dfu-util) и создать скрипты автоматической прошивки.

Оборудование: STM32F103, ST-Link, OpenOCD.

Задание:

1. Установить OpenOCD (или stlink-tools).
2. Подключить программатор к МК и ПК.
3. Написать скрипт (bash или batch), который:
 - компилирует проект (вызов make);
 - подключается к МК через OpenOCD;
 - стирает flash;
 - записывает .hex или .bin файл;
 - запускает прошивку.
4. Проверить, что после запуска скрипта МК работает.

Методические указания:

- Пример команды OpenOCD: `openocd -f interface/stlink.cfg -f target/stm32f1x.cfg -c "program build/project.elf verify reset exit"`.
- Для Arduino: использовать `avrdude`.

Форма отчёта:

Листинг скрипта, вывод OpenOCD в консоли, видео процесса прошивки.

Контрольные вопросы:

1. Чем отличается прошивка через SWD от UART bootloader?
2. Как защитить прошивку от чтения (RDP)?
3. Что такое «верификация» после прошивки?

Практическая работа №6. Анализ программного кода для микроконтроллера – 4 часа

Цель: научиться использовать отладчик (GDB) и логирование для поиска ошибок в прошивке.

Оборудование: STM32CubeIDE (встроенный отладчик), плата.

Задание:

1. Внести намеренную ошибку в код (например, деление на ноль, бесконечный цикл).
2. Запустить отладку (Debug), установить точку останова в подозрительном месте.

3. Использовать пошаговое выполнение (Step Into, Step Over), просмотреть значения переменных в окне Expressions.
4. Добавить логирование через UART (с уровнями DEBUG, INFO, ERROR).
5. Реализовать «сторожевой таймер» (IWDG), который перезагружает МК при зависании. Сымитировать зависание и проверить сброс.

Методические указания:

- Для IWDG: инициализировать таймер, сбрасывать его в основном цикле `HAL_IWDG_Refresh()`.
- Для логирования: макрос `#define DEBUG_PRINT(...) printf(__VA_ARGS__)`.

Форма отчёта:

Скриншоты отладки (точки останова, значения переменных), листинг логирования, осциллограмма сброса (или логи).

Контрольные вопросы:

1. Как отличить сброс от Watchdog от сброса по питанию?
2. Какие типы точек останова бывают (аппаратные, программные)?
3. Как анализировать дампы памяти после аварийного завершения?

Практическая работа №7. Написание приложения для опроса датчиков и отправки значений через MQTT – 4 часа

Цель: разработать программу для микроконтроллера (ESP8266 или STM32 + ESP-01), которая считывает данные с датчика (DHT22, DS18B20 или ультразвуковой) и отправляет их через MQTT на сервер (брокер Mosquitto или облачный).

Оборудование: плата NodeMCU (ESP8266) или STM32 + ESP-01, датчик DHT22, подтягивающие резисторы, Wi-Fi роутер, компьютер с MQTT брокером (Mosquitto) или публичный брокер (test.mosquitto.org).

Задание:

1. Для ESP8266 (NodeMCU):
 - Подключить датчик DHT22 к пину D4.
 - Написать программу на Arduino IDE, использующую библиотеки `ESP8266WiFi`, `PubSubClient`, `DHT`.
 - Подключиться к Wi-Fi сети (SSID, пароль заданы в коде).
 - Подключиться к MQTT-брокеру (например, `broker.emqx.io`, порт 1883).
 - Каждые 5 секунд считывать температуру и влажность, публиковать в топик `sensor/temperature` и `sensor/humidity` в формате JSON.
2. На компьютере подписаться на те же топики через MQTT-клиент (MQTT Explorer или `mosquitto_sub`), убедиться в приёме данных.
3. Добавить подписку на топик `cmd/led` для управления встроенным светодиодом (ON/OFF).

Методические указания:

- Пример MQTT publish: `client.publish("sensor/temp", String(temp).c_str());`;
- Для JSON: использовать `ArduinoJson` библиотеку.

Форма отчёта:

Схема подключения датчика, листинг программы, скриншот MQTT-клиента с полученными сообщениями, видео управления светодиодом.

Контрольные вопросы:

1. Какие QoS (Quality of Service) поддерживает MQTT?
2. Как обеспечить безопасность MQTT (TLS, аутентификация)?
3. Какой топик использовать для получения данных от всех датчиков одновременно?

4. КРИТЕРИИ ОЦЕНКИ ПРАКТИЧЕСКИХ РАБОТ

Оценка	Требования
5 (отлично)	Работа выполнена в полном объёме, все настройки корректны, программа функционирует без ошибок. Отчёт оформлен по требованиям, содержит скриншоты, листинги, выводы. Студент свободно отвечает на контрольные вопросы.
4 (хорошо)	Работа выполнена, но есть 1-2 недочёта (например, неверно настроен ACL, или в коде мелкая ошибка). На вопросы отвечает с небольшими затруднениями.
3 (удовлетворительно)	Работа выполнена частично (не менее 60%), основные функции работают, но есть существенные ошибки. Отчёт неполный.
2 (неудовлетворительно)	Работа не выполнена или не работает основная функция. Отсутствует отчёт. Студент не может объяснить свои действия.

5. ЛИТЕРАТУРА И ПРОГРАММНОЕ ОБЕСПЕЧЕНИЕ

Основные источники (из РП):

1. Зверева В.П., Назаров А.В. Сопровождение и обслуживание программного обеспечения компьютерных систем. – М.: Академия, 2020.
2. Федорова Г.Н. Разработка модулей программного обеспечения для компьютерных систем. – М.: Академия, 2020.
3. Гуров В.В. Микропроцессорные системы. – М.: ИНФРА-М, 2022.
4. Огнева М.В., Кудрина Е.В. Программирование на языке C++. – М.: Юрайт, 2021.
5. Самуйлов К.Е. и др. Сети и телекоммуникации. – М.: Юрайт, 2021.
6. Соколова В.В. Разработка мобильных приложений. – М.: Юрайт, 2023.

Программное обеспечение:

- Cisco Packet Tracer (бесплатно для учебных заведений)
- VirtualBox, Astra Linux (образ)
- STM32CubeIDE, Arduino IDE
- Git, Cppcheck, OpenOCD
- MQTT Explorer, Mosquitto
- Yandex Cloud / Google Cloud (пробный период)